REMARKS

Reconsideration of this application is respectfully requested in view of the foregoing amendment and the following remarks.

Claim 1 was pending in this application. Claims 2-19 have been added, no claims have been amended or cancelled. Accordingly, claims 1-19 will be pending herein upon entry of this Amendment. Support for the new claims can be found in the Specification and Drawings as originally filed. For the reasons stated below, Applicant respectfully submits that all claims pending in this application are in condition for allowance.

Objections to the Drawings

The objections to the Drawings have been obviated by the amendments to the Specification presented above.

Objections to the Specification

The Examiner has objected to paragraph 17 of the Specification as allegedly including an embedded hyperlink and/or other form of browser-executable code. See the 7/12/2005 Office Action at page 4. However, paragraph 17 includes a URL (specifically www.phrack.com/archive), and not a hyperlink. Applicants refer the Examiner to M.P.E.P. § 608.01, which does not preclude the inclusion of URL's (i.e., "web site addresses") in the Specification.

The Examiner has made various other objections with respect to the Specification. These additional objections are moot in light of the amendments to the Specification presented above.

Claim Rejections

Claim 1 stands rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Wagner et al. (NPL A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities), and further in view of Viega et al. (NPL ITS4: A static Vulnerability Scanner for C and C++ Code). This rejection is traversed because the cited references, both alone and in combination, fail to teach or suggest all of the features of the claimed invention.

More particularly, independent claim 1 recites, among other things, comparing the abstract syntax tree and the classes of known software vulnerabilities to identify a set of potential exploitable software vulnerabilities. In an exemplary embodiment, a vulnerability code analyzer uses information regarding various fault classes from a knowledge database to scan an abstract syntax tree of software code to flag vulnerabilities in the code. See the Specification at paragraph 41.

In contrast, Wagner appears to teach a method of static analysis of computer code that includes defining a constraint language, generating constraints from the computer code using the mathematical foundation of the defined constraint language, solving the resulting constraint system, and checking all of the string variables for overflow. See Wagner at § 1.1. The Examiner alleges that Wagner suggests comparing an abstract syntax tree and classes of known software vulnerabilities to identify a set of potential exploitable software vulnerabilities at § 1. See the 7/12/2005 Office Action at page 5. But § 1 of Wagner is merely an introduction and overview of the method of Wagner, which detects potential software vulnerabilities through mathematical calculation, and not by comparing an abstract syntax tree with classes of known software vulnerabilities. Therefore, Wagner does not disclose comparing the abstract syntax tree and the classes of known software vulnerabilities to identify a set of potential exploitable software vulnerabilities.

The Examiner acknowledges that Wagner is deficient at least for failing to teach creating a vulnerability knowledge database comprising one or more classes of known software vulnerabilities. *See* the 7/12/2005 Office Action at page 5. The Examiner maintains that Viega teaches creating a vulnerability knowledge database comprising one or more classes of known software vulnerabilities, and that it would have been obvious to include the vulnerability database of Viega "because it was well known in the art to use a database of vulnerabilities to check source code against it." *See id.* at pages 5 and 6. Assuming arguendo that the Examiner's characterization of Viega and statement of motivation were proper, which they are not, Viega does not cure the deficiency of Wagner discussed above. For at least these reasons, the rejection of claim 1 is improper and should be withdrawn.

Newly Added Claims

Claims 2-19 are newly added by this Amendment. Claims 2-5 depend from claim 1, and therefore, are allowable based on their dependency as well as for the features that they add to the claim 1. Independent claims 6 and 12 recite, among other things, features similar to those discussed above with respect to claim 1. Accordingly, claims 6 and 12 are allowable over Wagner and Viega for at least the reasons set forth above. Claims 7-11 and 13-19 depend from corresponding ones of independent claims 6 and 12, and therefore are allowable based on their dependency, as well as for the features that they add to the independent claims.

Conclusion

In view of the foregoing all of the claims in this case are believed to be in condition for allowance. Should the Examiner have any questions or determine that any further action is desirable to place this application in even better condition for issue, the Examiner is encouraged to telephone applicants' undersigned representative at the number listed below.

PILLSBURY WINTHROP SHAW PITTMAN LLP
1650 Tysons Boulevard
McLean, VA 22102
Tel: 703/770-7900

Date: January 20, 2006

Respectfully submitted,

By: [illegible]
Lawrence D. Eisen
Registration No. 41,009

Attachment: Abstract (marked up and clean version)

Customer No. 28970

OIPE IAP13 JAN 2 0 2006 PATENT & TRADEMARK OFFICE

ABSTRACT

~~The present invention provides a methodology~~ A system and method for certifying software for essential and security-critical systems. The system and method provide a methodology and corresponding analysis engines increase the level of confidence that common vulnerabilities are not present in a particular application. A pipeline system consisting of independent modules which involve increasingly complex analysis is disclosed. The pipeline approach allows the user to reduce computation time by focusing resources on only those code segments which were not eliminated previously in the pipeline. ~~The first step of the pipeline consists of flagging all potential vulnerabilities contained in an extendible vulnerability knowledge database (VKdb). This stage then filters vulnerabilities based on context information and passes them to the second stage. The second stage performs complex static analysis on the vulnerabilities and passes the remaining ones to a dynamic analysis stage. The pipeline approach allows very effective use of computation time and is the basis for our software certification methodology.~~